A VLSI Processor for Fast Track Finding Based on Content Addressable Memories

A. Annovi, A. Bardi, M. Bitossi, S. Chiozzi, C. Damiani, M. Dell'Orso, P. Giannetti, P. Giovacchini, G. Marchiori, I. Pedron, M. Piendibene, L. Sartori, F. Schifano, F. Spinella, S. Torre, and R. Tripiccione

***Abstract*—The authors describe a VLSI processor for pattern recognition based on content addressable memory (CAM) architecture, optimized for on-line track finding in high-energy physics experiments. A large CAM bank stores all trajectories of interest and extracts the ones compatible with a given event. This task is naturally parallelized by a CAM architecture able to output identified trajectories, searching for matches on 96-bit wide patterns, in just a few 40-MHz clock cycles. We have developed this device (called the AMchip03 processor) for the silicon vertex trigger (SVT) upgrade at the Collider Detector experiment at Fermilab (CDF) using a standard-cell VLSI design methodology. This approach provides excellent pattern density, while sparing many of the complexities and risks associated to a full-custom design. The cost/performance ratio is better by well more than one order of magnitude than an FPGA-based design. This processor has a flexible and easily configurable structure that makes it suitable for applications in other experimental environments. They look forward to sharing this technology.**

***Index Terms*—Parallel processing, particle tracking, pattern matching, triggering, very large scale integration (VLSI).**

I. INTRODUCTION

TRACK reconstruction in high energy physics experiments requires large online computing power. A conceptually very simple and effective approach to the problem, proposed several years ago [1], can be applied to a large class of detectors, e.g., typical vertex detectors, that can be conceived as segmented in layers, each layer being in turn divided into a number of bins. For each event a number of particle tracks traverse the detector. Each track crosses one bin per layer, generating hits. Tracks are, therefore, associated to specific strings of hits and misses (that we code as 1 and 0s, respectively). All tracks of physical interest correspond to bit patterns that are explicitly enumerated and stored in an appropriate data bank. Track reconstruction in this approach amounts to retrieving from the data bank the string (or strings) that match the current event, as measured in the detector. This task can be performed with negligible time delay by a content addressable memory (CAM), i.e., a device that compares in parallel an input pattern set of hits with all stored patterns and return the address(es) of the matching location(s).

A full-custom VLSI technology was used in this context to produce the first CAM for the CDF experiment [2], where tracks inside very high multiplicity events must be found in a time span of a few microseconds. A fast online tracker for the CDF silicon vertex trigger (SVT) [3] was built around the purposely developed associative memory (AM) chip [4], and operated in CDF since Run II started in the year 2000.

A critical figure of merit for a CAM-based track reconstruction system is the number of patterns that can be stored in the data bank. In the past, the request to maximize available patterns forced a full-custom VLSI approach, which implied a big development effort and a difficult upgrade path to more recent and denser microelectronic technologies, as they eventually become available. In recent years, very high-density silicon technologies make it possible to build a very large number of transistors inside a reasonably large silicon area (say, $\sim 1\ \text{cm}^2$). It is therefore appropriate to reconsider the best tradeoff between pattern density and ease of design (and eventually redesign). While the full-custom approach obviously maximizes pattern density, an FPGA-based design gives the fastest development time at the cost of a drastically reduced pattern density. This option has been considered in [5]. Despite the recent FPGA progress, these devices are still not convenient for our application. Midway between the two approaches, a standard-cell based design brings substantial advantages, as discussed in details later on.

In this paper, we describe the design and test of a new much more powerful version of the AM chip, that we call the AMchip03 processor, now used for the upgrade of the SVT at CDF. The AMchip03 uses a 0.18-μm CMOS technology and a strictly standard-cell based VLSI design approach.

The SVT processing time is a function of the detector occupancy and increases as the instantaneous luminosity rises. The AM chip upgrade increases the number of patterns stored in the chip from 128 to 5×10^3. A larger bank is available to store higher resolution candidate tracks, drastically reducing the number of found fakes and their processing time [6]. The number of chips per board has not changed [7]. The advantages and first results of this upgrade are described in detail [6].

II. AMCHIP03 ARCHITECTURE

The AMchip03 is a single chip digital VLSI processor performing all functions needed for CAM-based fast track identifi-

Manuscript received November 10, 2005; revised March 18, 2006. This work was supported by INFN.

A. Annovi and S. Torre are with INFN, Laboratori Nazionali di Frascati, I-00044 Frascati, Rome, Italy.

M. Bitossi and M. Dell'Orso are with INFN Sezione di Pisa, 56127 Pisa, Italy. They are also with Dipartimento di Fisica, Università di Pisa, 56127 Pisa, Italy.

A. Bardi, P. Giannetti, M. Piendibene, and F. Spinella are with INFN Sezione di Pisa, 56127 Pisa, Italy (e-mail: paola.giannetti@pi.infn.it).

S. Chiozzi, C. Damiani, F. Schifano, and R. Tripiccione are with INFN Sezione di Ferrara, 44100 Ferrara, Italy. They are also with Dipartimento di Fisica, Universita' di Ferrara, 44100 Ferrara, Italy.

P. Giovacchini is with STMicroelectronics Srl, 20041 Agrate, Milano, Italy.

G. Marchiori is with Atmel Roma s.r.l., 00155 Rome, Italy.

I. Pedron is with Dipartimento di Fisica, Universita' di Ferrara, 44100 Ferrara, Italy. He is also with Atmel Roma s.r.l., 00155 Rome, Italy.

L. Sartori is with INFN Sezione di Ferrara, 44100 Ferrara, Italy. He is also with Dipartimento di Fisica, Universita' di Ferrara, 44100 Ferrara, Italy. He is also with STMicroelectronics Srl, 20041 Agrate, Milano, Italy.

Digital Object Identifier 10.1109/TNS.2006.876052

cation and reconstruction. It formats and compresses the results and passes them to a daisy-chain of further AMchip03 components and eventually to the downstream stages of the data readout system.

As aforementioned, a large bank of precalculated patterns is used to find tracks in a detector consisting of a number of layers, each layer being segmented into a number of bins. For each physical event a number of tracks traverse the detector. Each track crosses one bin per layer, generating hits. The collection of tracks creates a particular configuration of hits: we use the word event when referring to the full set of hit bits. We know which bins have been hit and from this information we want to reconstruct the trajectories of all particles.

To this purpose we list offline all the relevant tracks that can go through the detector. A pattern, consisting of one hit bin per layer, corresponds to each possible track. All the different patterns are stored in a sufficiently large memory with CAM capabilities, that we call the pattern bank.

In principle, the pattern bank may contain all possible tracks that go through the detector (a 100% efficient bank). In practice, one should also consider effects which make particles deviate from the ideal trajectory, such as detector resolution smearing, multiple scattering, etc. Those effects generate a huge number of extremely improbable patterns, which blow the bank size up. For this reason, we decide to use a bank that is partially inefficient. We generate tracks in the detector and we convert tracks into patterns. We store new patterns corresponding to the generated tracks, until the bank reaches the desired efficiency. This procedure automatically ensures that high-probability patterns are stored and low-probability patterns are left out.

The generated track typology also affects the bank size. It is very convenient to restrict the range of the generated track parameters, such as PT and the region where they come from (luminosity region), to those values relevant for the physical processes to be studied.

At CDF, tracks are stored in the bank if their PT is larger than 2 GeV and if pointing to a source region in the transverse plane, few millimeters wide around the beam spot. This restriction helps to keep the size of the pattern bank small, but reduces to zero the efficiency for tracks coming from long lived particles. However, B-meson decay products, whose impact parameters are few hundred microns, are compatible with such a luminosity region and pattern bank.

When the beam moves in the transverse plane more than 300 μm, the pattern bank looses efficiency uniformity, thus, it is generated again, centering the new beam spot position. This happens roughly once a month.

A few details about the generation of the bank could be useful. For CDF applications we use simulation to generate tracks in the detector and then we convert them into patterns. The pattern bank is always generated to take advantage of all the patterns available in the hardware. The pattern recognition resolution, i.e., geometrical size of the patterns, is tuned to get a reasonably high pattern bank efficiency ($\sim$97%) using all patterns.

For each event one has to scan the pattern bank and compare each pattern to the event. A track candidate is found whenever all the hits in the pattern, or a majority of them, are present in the event. The minimum number of layers that have to be hit for a track to be found is a free parameter and is set in a control register as a single programmable threshold common to all patterns in the chip. Threshold comparison is introduced to account for any inefficiency in the detectors. A typical configuration is five layers out of six. The CAM technology, that allows performing this search in just a few clock cycles, is obviously the key component of the AMchip03 (in the following we will also loosely refer to the CAM bank with the term "Associative Memory").

It must be emphasized that with respect to commercially available CAMs the AMchip03 has the unique ability to search for correlations among input words received at different clock cycles. This is essential for tracking applications since the input words are the detector hits arriving from different layers. They arrive at the chip, serialized on six buses, without any specific timing correlation. Each pattern has to keep memory of each fired layer (the layer-match flip flop described below) until the pattern is matched or the event is fully processed and thus patterns can be reset.

The AM allows comparing at the same time all the previously stored patterns with the current event. The AMchip03 performs pattern recognition in a detector of up to 6 layers. Each pattern is therefore segmented into six 16 bit words, one for each detector layer. Fig. 1 displays a block diagram of the complete device. The chip receives 6 buses (Bus0-5 on the left of the figure) bringing hits from the detector, one bus for each layer. The buses received by the chip are 18 bit wide, but the two most significant data bits are matched with a mask once per chip and not distributed to the patterns. The mask is downloaded in a control register at initialization time. At each clock cycle the content of each pattern word is compared in parallel with the content of the corresponding bus and a flip-flop is set for each occurring layer match. When the number of layer-matches for a certain pattern is above a given threshold, a pattern-match flip-flop is set. Threshold comparison is introduced to account for any inefficiency in the detectors. It is possible to require that one specific layer has fired, in order to set the match flip-flop. The pattern bank uses approximately 80% of the silicon resources in the device and contains 5120 patterns, corresponding altogether to approximately 500 000 content-addressable memory bits.

In addition to the default 6 layer mode, the chip can be configured to perform pattern recognition for a detector of up to 12 layers, combining pairs of 6-word patterns into a 12–word pattern. In this case, the eighteenth bit of each data bus is used to distinguish hits coming from two layers, multiplexed on a single bus.

The processor contains the logic functionalities needed to interface with other sections of the readout system. The addresses identifying all matched pattern are read-out from the chip through the Patt_add_out bus. Patterns are read-out one after the other, giving priority to the highest address. When a pattern is read-out, the "kill" logic resets the "pattern-match" flip flop until the end of the event, so each matched pattern is read-out once.

Several AMchip03 processors can be cascaded in daisy chain to build a larger pattern bank. Multiple daisy chains can in turn be multiplexed for even larger banks. Each AMchip03 outputs its matched addresses on the Patt_add_out bus. Pattern addresses can be received on the Patt_add_in bus by the following AM-

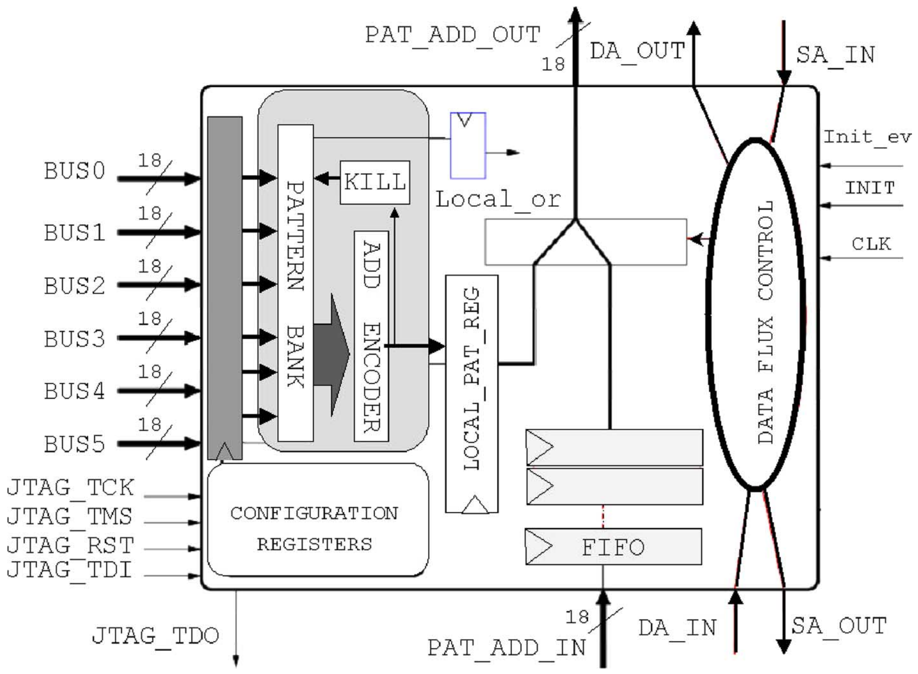

Fig. 1. Block diagram of AMchip03. The pattern bank, the readout and the control logic are emphasized (see text for description). (Color version available online at http://ieeexplore.ieee.org.)

chip03 in the pipeline, unless the chip is programmed to be the first in the chain. These addresses are stored in a FIFO and multiplexed with local addresses on the only output, Patt_add_out. The flux of addresses is controlled by a finite-state-machine (Data Flux Control in the figure) that uses a simple protocol based on four control signals, labeled as DA_out, DA_in SA_out and SA_in in the figure.

The core of our architecture is similar to the old AM device, previously used in CDF [4], but it has a much larger array size. On the other hand, this design substantially improves input-output capabilities, critical to reduce processing time. A short processing time is an important asset, since it helps to reduce the trigger dead-time. The time associated to actual pattern matching is negligible once data is available inside the chip, since the CAM array identifies a matching pattern in just one clock cycle. Hence, total processing time is determined essentially by the input bandwidth. In CDF, 12 independent systems, each made of an array of 128 AMchip03, handle the 12 Silicon detector wedges of the experiment. Aproximately 300 hits (at the accelerator luminosity 3×10^{32} cm^{-1} s^{-1}) are expected on average from the most busy wedge. Due to system constraints, all hits are loaded into the chip over one bus (40 MHz rate) at a time, so input time is $\sim$7.5 μs. Processing is already active as data is input to the processor, and matched patterns are immediately offloaded, with the last pattern coming out approximately six clock cycles (150 ns) after the last input data is received. In a more demanding environment (we consider a typical low luminosity situation at the Large Hadron Collider at CERN), with $\sim$30 000 hits in the whole detector, we might expect $\sim$2000 hits going to one AMchip03 array handling for instance $1/16$th of the whole detector. Exploiting the full input bandwidth (6 buses), input time in this case would be of the order of 8 μs.

III. Implementation and Tests

A. Design, Placement, and Routing

The AMchip03 has been logically designed using the VHDL hardware description language, and has been mapped on a standard-cell library using Synopsis tools for VLSI design. The 0.18-μm CMOS process (with 1 poly and 6 metal (Al) layers), available from the silicon foundry UMC, has been chosen. The automatically synthesized pattern array has been then further optimized manually, trying to reduce the logic complexity and with an eye to regular placement and routing.

The CAM bank is the part of the project that occupies most of the area so it is important to exploit as much as possible the intrinsic regularity of the array. The main goal at this stage is gate density, since our speed goals (40–50 MHz clock frequency) are easily obtained with the available silicon technology. Only a manual study of the structure can guarantee this result. Once an array structure reasonably close to optimal is obtained with a few trial and error steps, the associated cell placement is coded into a relatively simple Perl script. The remaining logic has been handled by standard software tools. Synopsys Physical-Compiler was used for placement of all other processor modules and Cadence First-Encounter for routing. The die size is approximately 9.8×9.8 mm^2. The chip core uses approximately 80% of the available area. The chip has a total of 164 logical input and output pads and is packaged in a standard 208-pin plastic "flat-pack." Fig. 2 shows the micrograph layout of the AMchip03.

B. Design Verification

We come to the common problem of chip design functionality verification against specification. This problem was separated into two logical steps. First take advantage of the fact that

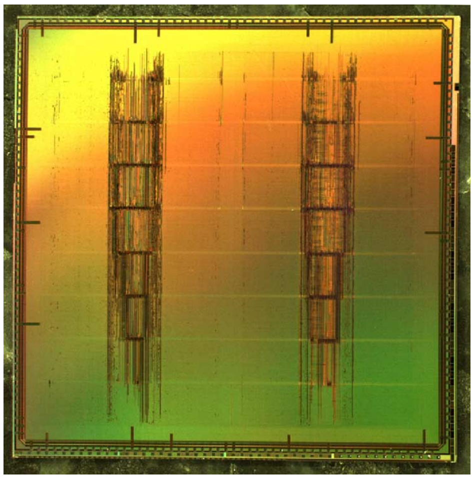
Fig. 2. Micrograph of the AMchip03 device. Four manually optimized columns of 1280 patterns each are visible. One on the left, one on the right and two in the middle. The two columns of lower density logic correspond to the interconnection and readout logic that was automatically placed. (Color version available online at http://ieeexplore.ieee.org.)

standard-cell designs are described by detailed and reliable Verilog models to be used for simulation. Second compare device simulation with appropriate reference test vectors that comply with, and verify, functional specifications. In order to prepare the reference test vectors, we independently developed a C-language model of the AMchip03. We wrote a total of 200 Kbytes of code, this coding work was done in parallel with chip design. Two students performed this work in about six months. A set of input test vectors was randomly generated, in order to load random patterns and random input data. Another set is made of manually defined test vectors used to test all the specific configurations.

Both the C and Verilog model have been extensively simulated with those test vectors prior to fabrication. Finally the outputs of the two models have been compared and debugged in order to understand and remove any discrepancies. At the end of this process, we had a fully debugged design, a clarification of all misinterpretations of the specifications as well as a working device with the first prototype. Post-routing simulations and static timing analysis shows a maximum operating frequency marginally higher than 50 MHz in worst case conditions.

C. *Production and Test*

Prototypes of the device were fabricated in a multiproject chip (MPC). A pilot production run of 12 wafers ($\sim$3000 chips) with dedicated masks was performed immediately after. The measured yield on the MPC has been disappointingly low ($\sim$37%). We got an additional 30% chips that were “almost good”: everything being as expected apart for a single bit in a single pattern inside the memory bank. The failing bit was not in a particular position, but was randomly distributed all around the bank area, suggesting the existence of defects randomly distributed. The dedicated fabrication run, chosen for the production, had a yield ($\sim$70%) marginally better than expected for our die size ($\sim$68%). The expected yield obtained with the pilot run suggests that the parameters chosen for the MPC process are not appropriate for high-density, manually optimized memories as the pattern bank. The first pilot run was sufficient to produce the number of chips necessary for the CDF upgrade ($\sim$2000 chips).

We have developed a test board with a zero insertion force (ZIF) socket to test the processor before soldering it on the application boards. The test board connects to a Pattern Generator and Digital Analyzer. FIFOs are used to handle the large amount of input data (6 hit buses, 18 bits for each bus). The prototype standard cell chips have been extensively tested with the same test vectors used to validate the design trough simulation. Software to convert automatically the test vectors from the simulation format to the test stand format environment has been developed. Chips selected by the test stand have been soldered on boards and tested on the experiment, showing that the selection performed in the single chip test stand is 100% efficient. Chips have been tested up to a frequency of 40 MHz, the maximum input frequency required by SVT. A higher frequency would require a larger power without any real advantage because other boards would be the bottleneck in the SVT pipeline.

The AMchip03 pinout is compatible with an FPGA chip, so that extensive board tests have been prepared before receiving our prototype. We have used a commercial low cost FPGA family (Xilinx Spartan 0.35-μm process) [8]. The FPGA-based AMchip03 has been logically designed with the same VHDL code that defines the real device. As expected from the discussion above, the pattern count in the FPGA is drastically reduced with respect to the full AMchip03 implementation (only 2 patterns, instead of 5120).

For SVT, 9U VME boards housing four mezzanines are used. Each board houses up to 128 chips, corresponding to 640K patterns [7], although only 64 chips are used in the currently available system and only 4000 patterns are stored in each chip. The full system has 24 boards (that is, 1536 AMchip03 chips), handling the 12 wedges of the detector. In conclusion the approved SVT upgrade was planned and built aiming at 512 000 patterns per wedge, even if the capacity of the two installed boards per wedge is 1.6×10^6 patterns. A further extension is possible in the future, adding more chips on the boards and improving other SVT boards [6].

The AMchip03 implements a Jtag interface. This interface gives write and read access to internal configuration of the AMchip which includes pattern memory, internal registers and Boundary Scan logic. All initialization and book-keeping operations are performed via the JTAG interface. The 128 AM chips on a board are interconnected on separate daisy-chains of 4 devices each, to build a total of 32 individual daisy-chains. The 32 separate chains are accessed in parallel by a VME slave which interfaces to all Jtag control signals of the chains. The VME 32-bit wide data transfer allows us to program the 32 chains in parallel, so the time needed to store all patterns in the chips is a few seconds. Patterns are also checked using the boundary scan. This operation takes a few minutes and is repeated at each run start, since it is shorter than other CDF initialization procedures executed in parallel. The bank is downloaded only if it is found to be corrupted or if it has to be changed because the beam spot has moved in the transverse plane.

The measured currents, Icore and I_{IO}, drawn by the AMchip03 from, respectively, 1.8 and 3.3 V power supplies are

TABLE I
POWER CONSUMPTION OF AMCHIP03

Frequency (MHz)	I_{core} (mA)	I_{IO} (mA)	Power (W)
10	200	40	0.5
20	400	80	1.0
40	700	160	1.8

shown in Table I. Measurements have been done in worst-case conditions when all input buses switch simultaneously.

D. Comparison With Other Technologies

It is interesting to compare the performances of our processor with other possible approaches. Let us start with a comparison with an FPGA-based implementation.

A serious problem related to FPGA is the package. Very powerful chips come only with very high pin-count, that is large packages that decreases the total number of chips we can put on a single board. Packages of the right size for our application correspond to small FPGAs. The board has been developed for PQ208 packages [7] and the upgrade is really fast only if we can avoid doing the board again. Moreover full pin-compatibility is often ignored when a new powerful family becomes available.

The most critical resources for a CAM structure are storage elements that can be accessed all in parallel. In a typical FPGA, these storage elements can be easily implemented with random flip-flops. A typical advanced FPGA family (for instance, Altera Stratix 2 [9]) has between 12 000 and 140 000 flip-flops. Each patterns requires of the order of 100 flip-flops, so between ~100 and ~1500 patterns could be squeezed inside one FPGA, under the very optimistic assumption that 100% usage as well as satisfactory speed can be obtained at the same time. Taking the largest FPGA option as an example the overall chip inventory (and the corresponding engineering problems) would increase by a factor 3 to 4, and the overall system costs by an unbearable factor of 30 to 40. The design effort would be probably reduced by a small factor from the eight months needed for the development of the AMchip03.

The use of distributed small RAMs contained inside the CLBs can improve the FPGA pattern capacity, as shown in [5]. However the gain is dependent on the CLB architecture and the kind of project is chosen. In that design [5], for example, the logic and buses to read/write patterns was not necessary for a specific feature (disappeared in the following release) of the development software available at that time. A careful, time-consuming job is necessary to evaluate and choose the right FPGA, reducing the main advantage the FPGA offers: a short project development time.

At the other technological extreme, a full custom VLSI approach (at least for the regular CAM array) might increase density by a factor of approximately 4, while fabrication costs would stay almost constant. On the other hand the design process would be much longer in time, intensive logic simulation, timing verification before prototyping would be difficult, and re-targeting to a different VLSI technology would be almost equivalent to a full re-design.

Finally, it is interesting to estimate the hardware complexity of a farm of PCs trying to perform our pattern matching task in software with comparable performance. Direct implementation in software of the actual algorithm performed by the Amchip03 is obviously ruled out. In fact, measuring the timing of a logical simulation of the 1 500 AMchip03 installed in CDF we find that we would need roughly 100 000 Pentium 4 CPUs in order to sustain the 25 kHz event rate.

More efficient algorithms are available however. Let us focus first on just one layer of w bits (that is $w = 12$ bits for CDF). On average, if $N = 640\ 000$ patterns (again a typical CDF value for a single detector wedge) are recorded in the data-base, one specific single layer hit will be shared by $N/2^w = 160$ different candidate patterns. We can arrange this information in precomputed lists, indexed, for the given layer, by the hit. The list includes all candidate patterns that share that hit.

One such list must be prepared for each of the L layers. For each incoming hit the corresponding list has to be visited, to set as "fired" the corresponding layer into all the candidate patterns of the list. After all hits are received (approximately $n = 50$ per layer) we end up with an overall list of $n \times N/2^w$ entries checked by the algorithm. At this point the algorithm, must inspects the L lists for matches that, in this case, are associated to the same pattern being present in all L lists (or in a majority of them). Efficient algorithms exists to perform the task, with a number of memory accesses of the order of

$$n\frac{N}{2^w}\log_2\left(n\frac{N}{2^w}\right). \tag{1}$$

This step must be performed L times, implying a number of memory accesses of the order of

$$KLn\frac{N}{2^w}\log_2\left(n\frac{N}{2^w}\right) \tag{2}$$

where K is a small integer $K = 2\ldots5$.

Using CDF reference values, we end up with approximately 3×10^6 accesses. If we optimistically assume that all data is available in a large second level cache (typical access time optimistically estimated at 2 ns) and just neglect any processing overhead, we conclude that approximately 300 PCs could perform the task in parallel in order to stay within the allotted time limits. A system able to handle all 12 wedges would therefore include from 3000 to 4000 PCs.

IV. Conclusion

A track finding processor based on a CAM has been produced for the SVT upgrade of the CDF experiment. This device improves the size of the pattern bank by a factor ~40 over a previous version of the processor. The device is also faster: it was designed with a target clock frequency of 50 MHz (although it is currently operated in a 40 MHz system).

The design uses a standard-cell based technology instrumental to keep the design time to about 8 months (plus 2 months to fabricate the prototypes). Our design methodology makes upgrades to new technologies or to different environments straightforward and quick to implement.

We can extrapolate the future capacity of a new device based on 130-90 nm technologies: we expect an improvement of at least a factor 2–4 in pattern density. A good strategy is to use FPGAs for system development, and to start migration to a pin compatible standard-cell device only for system production when the experiment is getting close to data taking. Clock frequencies, that in the present application had very relaxed requirements, could also be increased by large factors (~ 5). This might provide opportunities to use these devices in applications with tighter time constraints, such as Level 1 trigger systems.

ACKNOWLEDGMENT

The authors would like to thank C. Das, P. Malisse, and D. Lambricht of IMEC (Belgium) and G. D'Amelio of Microtest (Italy) for their precious collaboration in all production and test phases of the AMchip03.

REFERENCES

[1] M. Dell'Orso and L. Ristori, "VLSI structures for track finding," *Nucl. Instrum. Methods*, vol. A278, pp. 436–440, 1989.
[2] F. Abe, "The CDF detector: An overview," *Nucl. Instrum. Methods*, vol. A271, pp. 387–403, 1988, CDF Collaboration.
[3] S. Belforte, "SVT: An online silicon vertex tracker for the CDF upgrade," *Nucl. Intsrum. Methods*, vol. A409, pp. 658–661, 1998.
[4] R. Amendolia, "The AMchip: A full-custom MOS VLSI associative memory for pattern recognition," *IEEE Trans. Nucl. Sci.*, vol. 39, pp. 795–797, 1992.
[5] A. Bardi, "A programmable associative memory for track finding," *Nucl. Intsrum. Methods*, vol. A413/2-3, pp. 367–373, 1998.
[6] J. Adelman *et al.*, "First steps in the silicon vertex trigger upgrade at CDF," in *Proc. 2005 IEEE Nucl. Sci. Symp. Conf. Rec.*, Oct. 23–29, 2005, vol. 1, pp. 603–607.
[7] A. Annovi, "The AM++ board for the silicon vertex tracker upgrade at CDF," in *Nucl. Sci. Symp. Conf. Rec., 2005 IEEE*, Oct. 23–29, 2005, vol. 1, pp. 598–602.
[8] , [Online]. Available: http://www.xilinx.com
[9] *Altera Corp., Stratix II Device Handbook (2004)*, [Online]. Available: http://www.altera.co